RECEIVED
CENTRAL FAX CENTER
APR 0 9 2008

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/773,673
Filing Date	Feb 6, 2004
First Named Inventor	Pechanek, Gerald George
Art Unit	2183
Examiner Name	Johnson, Brian P.
Attorney Docket Number	800.0141
Total Number of Pages in This Submission	37

ENCLOSURES *(Check all that apply)*

- [ ] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment / Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) ________
  - [ ] Landscape Table on CD
- [ ] After Allowance communication to (TC)
- [ ] Appeal Communication to Board of Appeals and Interferences
- [X] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [X] Other Enclosure(s) (please identify below):

PTO 2038 Credit Card Form

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Priest & Goldstein, PLLC		
Signature	[illegible]		
Printed name	Peter H. Priest		
Date	April 9, 2008	Reg. No.	30210

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted 571-273-8300 to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Signature	Marianna Tortorelli		
Typed or printed name	Marianna Tortorelli	Date	April 9, 2008

RECEIVED
CENTRAL FAX CENTER
APR 09 2008

800.0141
A01214C1

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of	:	Pechanek et al.
For	:	Methods and Apparatus for General Deferred Execution Processors
Serial No.	:	10/773,673
Filed	:	02/06/2004
Group	:	2183
Examiner	:	Johnson, Brian P.

Durham, North Carolina
April 9, 2008

MAIL STOP APPEAL BRIEF – PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPELLANT'S BRIEF

Dear Sirs:

1. Transmitted herewith is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed on January 9, 2008.

2. The Applicant is other than a small entity.

3. Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is $510.00.

   [x] The Commissioner is hereby authorized to charge the fee of $510 our credit card.

   [x] The Commissioner is hereby authorized to charge the 1 month extension fee of $120 to our credit card. This letter petitions for a one month extension of time.

[X] The Commissioner is hereby authorized to charge any additional fees which may be required or credit any overpayment to Law Offices of Peter H. Priest Deposit Account No. 50-1058.

Respectfully submitted,

[illegible]

Peter H. Priest
Reg. No. 30,210
Priest & Goldstein, PLLC
5015 Southpark Drive, Suite 230
Durham, NC 27713
(919) 806-1600

RECEIVED
CENTRAL FAX CENTER

APR 0 9 2008

800.0141
A01214C1

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of	:	Pechanek et al.
For	:	Methods and Apparatus for General Deferred Execution Processors
Serial No.	:	10/773,673
Filed	:	02/06/2004
Group	:	2183
Examiner	:	Johnson, Brian P.

Durham, North Carolina
April 9, 2008

MAIL STOP APPEAL BRIEF – PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' BRIEF

Sir:

1. The Real Party In Interest

The real party in interest is the assignee, Altera Corporation.

2. Related Appeals and Interferences

None.

04/10/2008 HMARZI1 00000011 10773673
01 FC:1402 510.00 OP

04/10/2008 HMARZI1 00000011 10773673
02 FC:1251 120.00 OP

3. Status of the Claims

This is an appeal from the October 9, 2007 final rejection of claims 9-26, all of the pending claims. The title was objected to. Claims 9 and 18-26 were rejected under 35 U.S.C. § 102(e) based on Sheaffer U.S. Patent No. 6.957,321 (Sheaffer). Claims 10, 11, 13, and 15 were rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer in view of Moller U.S. Patent No. 6,826,522 (Moller). Claims 12, 14, and 16 were rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer and Moller in view of Tremblay U.S. Patent No. 6,341,348 (Tremblay). Claim 17 was rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer in view of Pechanek et al. U.S. Patent No. 6,173,389 (Pechanek). Pending claims 9-26 are the subject of this appeal.

4. Status of Amendments

The claims stand as last amended on August 21, 2006. No Amendment After-Final has been filed.

5. Summary of Claimed Subject Matter

The present invention relates generally to a very long instruction word (VLIW) memory system and processor apparatus. The VLIW memory system and processor apparatus supports standard size instruction words, such as 32-bit instruction words, and at least one extended format instruction word which has a format that is wider than a standard size instruction word, such as, a 40-bit or 64-bit instruction. The standard size instructions and extended format instructions may be grouped together to form a VLIW which may be indirectly accessed for execution, allowing the execution of the VLIW to be deferred until the execution is specified by a program. Each instruction that is stored in a VLIW instruction slot, including the extended format instructions, is specified in an instruction set architecture. Figs. 1-19 and the text at page 6 et seq. provide a detailed description of the invention.

Claim 9

More particularly, claim 9 addresses a "very long instruction word (VLIW) memory system". By way of example, a VLIW memory that is separate from a program instruction memory is used to store a plurality of VLIWs. Each VLIW is comprised of a set of instructions that are to be executed together in parallel. Instructions from the set of instructions are individually stored in instruction slots at a program specified address location that is used to locate the VLIW stored in the VLIW memory. Rather than store only fixed width instructions, the VLIW memory system of the present invention advantageously stores at least one expanded width instruction in an expanded width instruction slot. The format of an expanded width instruction is wider than that utilized by a standard width instruction. The VLIW memory is configured to load an expanded width instruction in the at least one expanded width instruction slot.

An exemplary VLIW memory system 500 shown in Fig. 5 and a suitable instruction set architecture are discussed at page 13, lines 7-21. Figs. 6A-6C and the text at page 14, line 21 – page 15, line 9 address a standard width instruction format and an extended width instruction format. Another exemplary VLIW memory system is shown in the deferred execution processor 2 (DXP2) processor organization of Figs. 12A and 12B and discussed at page 21, lines 16-22. The VLIW memory system of Figs. 12A and 12B may be adapted for use in conjunction with various embodiments of the present invention. An expanded instruction format with three exemplary instruction encodings 1700 is shown in Fig. 17 and discussed, for example, at page 26, line 18 – page 27, line 13 for use with a VLIW memory system, such as the VLIW memory system of Figs. 12A and 12B.

The VLIW memory system of claim 9 comprises a "VLIW memory having a plurality of instruction slots for storing VLIW instruction words at addressable locations in the VLIW memory from which VLIWs may be fetched for execution". The DXP2 processor organization of Figs. 12A and 12B illustrates a VLIW memory VIM basket 1210 having a plurality of instruction slots 1243 and 1244 for storing VLIW instruction words at addressable locations in the VLIW memory 1210. See present specification, page 21, lines 16-20. The VLIWs stored in the VLIW memory 1210 may be fetched for execution on decode and execution stages 1216 and 1218, respectively, as described, for example, at page 22, line 13 – page 23, line 1.

A typo was recognized at page 21, lines 19 and 20, in which the sentence "The VIMBasket 1210 consists of either four slots 1243 or six slots 1244 in Fig. 12 instead of the fixed five slot arrangement in Fig. 5" will be corrected by amendment to read "The VIMBasket 1210 consists of either four slots ~~1243~~ 1244 or six slots ~~1244~~ 1243 in Fig. 12 instead of the fixed five slot arrangement in Fig. 5". The amended text corresponds to the arrangement shown in Fig. 12B and is also supported by the text at page 22, lines 14 and 15. The proposed amendment to the paragraph beginning at page 21, line 16 is attached in an appendix titled "SPECIFICATION REPLACEMENT PAGE APPENDIX" found at page 34. Rather than submit this amendment at this time and possibly delay the processing of this case, it will be submitted upon resolution of other issues raised by this appeal.

The VLIW memory system of claim 9 also comprises "said plurality of instruction slots comprising standard width instruction slots for storing standard width instructions and at least one expanded width instruction slot having a width that is greater than the standard width instruction slots, the expanded width instruction slot used for storing an expanded width instruction having a format that is wider than the standard width instructions". The DXP2 processor organization of Figs. 12A and 12B illustrates the plurality of instruction slots 1243 and 1244 which comprise standard width instruction slots for storing standard width instructions, such as 32-bit instructions 1224 and 1226. The instruction slots 1243 in Fig. 12B, for example, show two slots for 64-bit instructions 1220 and 1222. An expanded instruction format with three exemplary instruction encodings 1700 is shown in Fig. 17 and discussed, for example, at page 26, line 18 – page 27, line 13 for use with a VLIW memory system, such as the VLIW memory system of Figs. 12A and 12B. A standard width 32-bit instruction format, such as prior art encoding 600 is shown in Fig. 6A.

The VLIW memory system of claim 9 further comprises the "VLIW memory configured for loading said at least one expanded width instruction slot with an expanded width instruction". One suitable configuration for the VLIW memory is the VIM basket 1210 shown in Fig. 12B. The VIM basket 1210 is configured for loading at least one expanded width instruction slot with an expanded width instruction through the use of a load VLIW (LV) instruction, such as, instruction LV1, that is specified by opcode 1115, shown in Fig. 11A and described, for example, at page 20, line 13 – page 21, line 4 and at page 23, lines 10-12.

Claim 17

Claim 17 addresses a "very long instruction word (VLIW) instruction memory basket (VIMB) system". By way of example, a VLIW instruction memory basket (VIMB) that is separate from a program instruction memory is used to store a plurality of VLIWs. Each VLIW is comprised of set of instructions that are to be executed together in parallel. Instructions from the set of instruction are individually stored in instruction slots at a program specified address location that is used to locate the VLIW stored in the VIMB. Rather than store only fixed width instructions, the VLIW memory system of the present invention advantageously stores at least one expanded instruction in an expanded instruction slot. The format of an expanded instruction is wider than that utilized by a standard instruction required in program storage. An instruction bit organizer is utilized to organize instructions received as data into a proper format for loading the instructions into the instruction slots, including loading at least one expanded instruction into the expanded instruction slot.

An exemplary VLIW instruction memory basket (VIMB) 510 organization 500 shown in Fig. 5 and a suitable instruction set architecture are discussed at page 13, lines 7-21. Figs. 6A-6C and the text at page 14, line 21 – page 15, line 9 address a standard width instruction format and an extended width instruction format. Another exemplary VIMB system is the load VIMB apparatus 700 utilized for loading a VIM Basket 710 and discussed at page 15, line 10 – page 16, line 13. The system 500 of Fig. 5 and the apparatus 700 of Fig. 7 may be adapted for used in conjunction with various embodiments of the present invention. A standard width instruction 600 suitable for use in program storage is shown in Fig. 6A and an expanded instruction having an exemplary encoding 610 for an expanded multiply accumulate unit (MAU) instruction is shown in Fig. 6B and discussed at page 14, line 21 – page 15, line 9, for example.

The VIMB system of claim 17 comprises an "instruction register for storing a load indirect VLIW (LIV) instruction comprising a load mask bit field specifying which instruction slots are to be loaded in a VLIW having at least one instruction slot that is an expanded instruction slot, the VLIW accessible to be loaded at an addressable location in the VIMB". The load VIMB apparatus 700 includes an instruction register (IR1) 712 for storing a LIV instruction comprising a load mask bit field 714 specifying which instruction slots are to be loaded at a location in the VIMB 710 as shown in Fig. 7A. The location of a VLIW in the VIMB 710 is specified by a VIM address that is based on bit fields of the LIV instruction, such as the VIMOFFS field bits 0-7 of IR1 712, shown in Fig. 7A. The instruction slots to be loaded have at least one instruction slot that is an expanded instruction slot, such as, the 36-bit MAU instruction slot shown in VIMB 710 in Fig. 7B and the 36-bit MAU instruction slot 518 shown in VIMB 510 in Fig. 5B. An encoded form of the MAU instruction 610 is shown in Fig. 6B for storage in the expanded MAU instruction slot 518 as an expansion by 8-bits over the storage required for a standard width MAU instruction as described, for example, at page 13, lines 7-15 and page 15, lines 7-9.

The VIMB system of claim 17 also comprises an "instruction bit organizer for receiving instructions as data and based on the load mask bit field organizing the bits from the data encoded instructions into proper format for loading into the specified instruction slots in the VIMB". Instruction bits are received over a data path as data encoded instructions and organized for loading into the VIMB 710 in an instruction bit distributor 720 as described at page 15, lines 10-18 and page 16, lines 6 and 7. Based on a load mask bit field 714, the instruction bit distributor 720 organizes the instruction bits received in the proper order for loading into specified slots in the VIMB 710 as described, for example, at page 15, line 22 – page 16, line 7.

The VIMB system of claim 17 further comprises the "a plurality of VLIWs from which VLIWs may be fetched for execution, wherein the width of the at least one expanded instruction slot is greater than the width of instructions required in program storage". Fig. 8A shows an exemplary 2x2 processor using a VIMB, such as VIMB 802 having a plurality of M

VLIWs each of J-bits, such as J=199 bits as shown in Figs. 5 and 7 and discussed at page 13, lines 11 and 12, page 15, lines 15-18, and page 16, lines 14-16. VLIWs are fetched from the VIMB, such as VIMB 510, for execution on execution units 530, 532, 534, 536, and 538 as shown in Fig. 5B and discussed at page 23, lines 18-21. The width of at least one expanded instruction slot, such as the expanded MAU instruction slot 518 of Fig. 5B, is 36-bits. This width is greater than the width of instructions required in program storage, such as the 32-bit encoding 610 of the MAU SUM2PA instruction shown in Fig. 6B and described at page 14, line 21 – page 15, line2, page 15, lines 7-9, and page 11, lines 1-6, for example.

Claim 18

Claim 18 addresses a "very long instruction word (VLIW) memory system". By way of example, a VLIW memory system utilizes a VLIW memory that is separate from an instruction memory. The VLIW memory is used to store a plurality of VLIWs and the instruction memory stores program instructions of a standard or first bit width. Each VLIW is comprised of a set of instructions that are to be executed together in parallel. Instructions from the set of instructions are individually stored in instruction slots at a program specified address that is used to locate the VLIW stored in the VLIW memory. Rather than store only fixed width instructions, the VLIW memory system of the present invention advantageously stores instructions of a second bit width in instruction slots also of the second bit width where the second bit width is different from the first bit width. An execute VLIW (XV) instruction when executed causes a VLIW to be fetched from an XV specified address for execution.

The VLIW memory system of claim 18 comprises an "instruction memory holding a plurality of instructions of a first bit width, the plurality of instructions having at least one execute VLIW instruction". A 32-bit instruction memory 105 of Fig. 3 stores 32-bit short instruction words as described at page 8, lines 10-13 and lines 17-19, where the first bit width is 32-bits. A short instruction word (SIW) fetch controller 103 fetches the 32-bit instructions from the instruction memory 105 and dispatches the 32-bit fetched instructions to other PEs in the system over a 32-bit instruction bus 102 as described at page 8, lines 17-19 and page 9, lines 2 and 3. For example, the SIW fetch controller 103 of Fig. 3 may cause a 32-bit execute VLIW (XV1) instruction residing in the instruction memory 105 to be fetched and distributed via an instruction bus path 1238 to be received in an instruction register, such as instruction register (Ir1) 1212 of Fig. 12A, as described at page 22, lines 5-9, for example.

The VLIW memory system of claim 18 also comprises a "very long instruction memory having instruction slots for storing instructions of a second bit width wherein the second bit width is different from the first bit width and wherein the very long instruction memory holds VLIWs at addressable locations that may be fetched as a result of executing the at least one execute VLIW instruction". A VLIW memory, such as VIM basket 1210 stores a VLIW in instruction slots, such as the four slots 1244 made up of 64-bit slots 1220 and 1222 and 32-bit slots 1224 and 1226, where the second bit width is, for example, 64-bits as described at page 22, lines 13-15. The XV1 instruction that was received in the IR1 1212 is then executed as illustrated in Fig. 13 which causes a VLIW to be fetched at a VIM basket address that is generated through an address adder circuitry 1242 as described, for example, at page 22, lines 8-17.

<u>Claim 20</u>

Claim 20 addresses a "very long instruction word (VLIW) memory system". By way of example, a VLIW memory system utilizes a VLIW memory that is separate from an instruction memory. The VLIW memory is used to store a plurality of VLIWs and the instruction memory stores program instructions of a standard width. Each VLIW is comprised of a set of instructions that are to be executed together in parallel. Instructions from the set of instructions are individually stored in instruction slots at a program specified address that is used to locate the VLIW stored at that specified address. Rather than store only fixed width instructions, the VLIW memory system of the present invention advantageously stores instructions of a compressed format in instruction slots also of the compressed format where the compressed format is narrower than the standard width. Means are provided for loading a compressed format instruction into a corresponding compressed format instruction slot.

The VLIW memory system of claim 20 comprises a "plurality of instruction slots for storing instruction words forming a VLIW, at least one of said plurality of instruction slots having a compressed format for storing a compressed instruction having a narrower instruction format with respect to an instruction format required in a program storage, wherein the VLIW resides at an addressable location in a VLIW memory, the VLIW memory holding VLIWs that may be fetched for execution". A VLIW memory, such as VIM basket (VIMB) 846 of Fig. 8B stores a VLIW in instruction slots, such as expanded instructions X0', Y0', Y1', Y2', Y3' and X1' 850, as described at page 17, lines 21-23. In a further example, arithmetic instructions may be specified to be encoded in either or both 15-bit and 32-bit SP instruction formats as shown in Fig. 9A with, for example, dual 15-bit instructions 914 and 916, as described at page 18, lines 14-19. The VLIW stored in the VIMB 846 may be fetched for execution on execution units 854 as described, for example, at page 17, line 21 – page 18, line 2.

The VLIW memory system of claim 20 further comprises "means for loading said at least one compressed format slot with a compressed instruction". Loading of at least one compressed format slot with a compressed instruction may be obtained through the use of a load VLIW 1 (LV1) instruction as specified by opcode 1115 of Fig. 11A and described, for example, at page 20, line 13 – page 21, line 4.

Claim 21

Claim 21 addresses a "processing apparatus". By way of example, a processing apparatus utilizes a memory that stores short instruction words (SIWs) that make up a processing apparatus program. The processing apparatus also utilizes an indirect very long instruction word (VLIW) memory that is separate from the SIW memory. The indirect VLIW memory stores a plurality of VLIWs and the SIW memory stores program SIWs where each SIW is of a fixed size. Each VLIW is comprised of a set of instructions that are to be executed together in parallel. Instructions from the set of instructions are individually stored in instruction slots at a program specified address that is used to locate the VLIW stored in the indirect VLIW memory. At least one of the instruction slots stores an instruction that advantageously has an instruction format sized according to execution function and operand storage capacity and independent of the size of the SIWs. The operand storage capacity is associated with at least one data memory which stores instruction operands. Also, at least two execution units are utilized for executing a VLIW fetched from the indirect VLIW memory. The VLIW is fetched and executed in response to a SIW, specified as an execute VLIW (XV) instruction, that is dispatched from the SIW memory.

The processor apparatus of claim 21 comprises a "memory for storing a processing apparatus program comprising short instruction words". A 32-bit instruction memory 105 of Fig. 3 stores 32-bit short instruction words as described at page 8, lines 10-13 and lines 17-19.

The processor apparatus of claim 21 comprises an "indirect very long instruction word (VLIW) memory comprising a plurality of instruction slots for storing instruction words, at least one of said instruction slots having an instruction format sized according to execution function and operand storage capacity and independent of the size of the short instruction words, wherein the plurality of instruction slots are organized to form a plurality of VLIWs and each VLIW resides at an addressable location in the indirect VLIW memory, the indirect VLIW memory holding VLIWs that may be fetched for execution". An exemplary VLIW instruction memory (VIM), such as, VIM basket (VIMB) 510, and the VIMB organization 500 is shown in Fig. 5. Each VLIW which resides at an addressable entry in the VIMB 510 is made up of a plurality of instruction slots, such as store unit instruction slots 512, load unit instruction slots 514, ALU, MAU, and DSU instruction slots 516, 518, and 520, respectively. A suitable instruction set architecture is utilized in which "all the instruction fields of the instructions stored in the VIMB can be expanded beyond a conformance to a fixed 32-bit instruction format", as discussed at page 13, lines 7-21. Figs. 6A-6C and the text at page 14, line 21 – page 15, line 9 address a standard width instruction format having a 32-bit encoding as shown in Fig. 6A and an extended width instruction format having a 40-bit encoding as shown in Fig. 6B. The extended width instruction format of Fig. 6B is sized independently of the size of the SIW format by, for example, using "2 additional bits in each operand field to extend the addressing range for the CRF" which supports a larger size compute register file than can be supported by the fixed size SIW format as shown in Fig. 6A. See also, present specification, page 13, lines 11-21 and page 14, lines 9-14. In addition, the opcode field of the extended width instruction format of Fig. 6B is expanded by 1-bit. Thus, the opcode field changes from a 6-bit field in the fixed format of Fig. 6A to a 7-bit field in the extended format of Fig. 6B to allow for more instruction types supporting additional execution functions, as described at page 13, lines 15-17 and page 14, lines 18-20. VLIWs are fetched from the indirect VLIW memory, such as VIMB 510, for execution on execution units 530, 532, 534, 536, and 538, as shown in Fig. 5 and discussed at page 23, lines 18-21, for example.

The processor apparatus of claim 21 also comprises "at least one data memory unit storing instruction operands". Instruction operands are stored in at least one data memory unit such as a processing element configurable register file 127 of Fig. 3 or one of the local data memories M0i 1420 or M1i 1422 of Fig. 14 as described at page 9, lines 17-19.

The processor apparatus of claim 21 further comprises "at least two execution units for executing a VLIW fetched from the indirect VLIW memory in response to a short instruction word dispatched from the memory". VLIWs are fetched from the indirect VLIW memory, such as VIMB 510, for execution on execution units 530, 532, 534, 536, and 538, as shown in Fig. 5 and discussed at page 23, lines 18-21. The VLIW stored in the indirect VLIW memory, such as the VIMB 510 or, for example, the VIMB 846, is executed in response to an execute VLIW (XV) SIW, such as the one depicted as stored in instruction register IR1 in Fig. 5A. The XV is distributed over an instruction bus as dispatched from the SIW memory as described with regard to Fig. 8B, for example, at page 17, line 13-21.

6. Grounds of Rejection to be Reviewed on Appeal

Claims 9 and 18-26 were rejected under 35 U.S.C. § 102(e) based on Sheaffer. Claims 10, 11, 13, and 15 were rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer in view of Moller. Claims 12, 14, and 16 were rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer and Moller in view of Tremblay. Claim 17 was rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer in view of Pechanek.

7. Argument

The final rejections under 35 U.S.C. §§ 102 and 103 did not follow the statutes and case law interpreting the statutes. For anticipation under 35 U.S.C. § 102, a single reference must teach each and every claim element of the claimed invention. Anticipation is not shown even if the differences between the claims and the prior art reference are insubstantial and could be supplied by the knowledge of one skilled in the art. Structural Rubber Products v. Park Rubber, 749 F.2d 707, 223 U.S.P.Q. 1264 (Fed. Cir. 1984). A feature not directly taught is only inherently present if a structure in the prior art necessarily functions in accordance with the claim. In re King, 801 F.2d 1324, 231 U.S.P.Q. 136 (Fed. Cir. 1986).

Under Section 103, the scope and content of the prior art are to be determined by the Examiner. Differences between the prior art and the claims are to be ascertained, and the level of ordinary skill in the art assessed. Secondary considerations may give light to the circumstances surrounding the origin of the subject matter to be patented. KSR International Co. v. Teleflex, Inc., 550 U.S. ____, ____, 82 U.S.P.Q. 2d 1385, 1391 (2007) (quoting and reaffirming Graham v. John Deere, 383 U.S. 1, 148 U.S.P.Q. 459 (1966)).

As will be illustrated below, the claims of the present invention are not anticipated and are not obvious in view of the references relied upon by the Official Action.

A. Section 102(e) Rejections

Claim 9

Claim 9 was rejected under 35 U.S.C. §102(e) as being anticipated by Sheaffer. Sheaffer describes an instruction set extension using operand bearing no operation (NOP) instructions as a way of extending the format of an individual instruction of an existing instruction set architecture. An instruction of an existing format in a program instruction stream is extended after a decode operation detects an associated operand bearing NOP instruction and thereby forms an expanded instruction by associating a NOP operand with one or more instructions. Both an existing instruction and its associated operand bearing NOP instruction must be decoded together before the benefits of the operand bearing NOP instructions can be obtained. This arrangement is explicitly shown in Sheaffer in Fig. 2B which shows "a process that may be carried out by the apparatus of Fig. 2a". Sheaffer, col. 4, lines 60-64. In Fig. 2B, block 225 states "receive first instruction <u>and</u> operand specifying NOP instruction" which as "indicated in block 225 of Fig. 2b, a first instruction 140 (Opcode A) and an operand specifying NOP instruction 150 are received by a decode module 205" as described at col. 4, lines 64-66. In Fig. 2B, block 235 states "<u>Associate</u> NOP operand(s) with first instruction and/or other instruction(s)" which as "indicated in block 235, the decode module 205 associates NOP operands with the first instruction 140" as described at col. 6, lines 14 and 15. Sheaffer continues, "[s]ubsequent to decoding by the decode module 205, the first opcode 140 (Opcode A) may be represented by a different opcode, Opcode B" as described at col. 6, lines 15-18 , In Fig. 2B, the next block 245 states "Perform operation specified by first instruction using operands specified by NOP instruction" which as "indicated in block 245, the new Opcode B is executed by an execution module 215 to perform an operation indicated by Opcode A on some or all of the operands from Opcode A and some or all of the operands from the NOP 150" as described at col. 6, lines 29. Thus, as described by Sheaffer, two separate instructions are involved, a first instruction defined by Opcode A 140 and a second instruction defined by the NOP instruction 150. It is only after Sheaffer decodes the two separate instructions that a new "Opcode B" is formed which is able to make use of the additional operands that are supplied by the operand bearing NOP instruction 150.

In contrast to Sheaffer, the present invention does not require a decoder, such as Sheaffer's decoder 205, to have the features and benefits of "at least one expanded width instruction slot having a width that is greater than the standard width instruction slots, the expanded width instruction slot used for storing an expanded width instruction having a format that is wider than the standard width instructions" as claimed in claim 9. In the present invention, the at least one expanded width instruction slot is loaded with an expanded width instruction. A VLIW with the loaded expanded width instruction that is stored in a VLIW memory may then be fetched for execution. By contrast, Sheaffer must fetch two separate instructions and then combine them in a decoder before the two separate instructions can be associated and then executed.

The Official Action suggests that Sheaffer's "first instruction and operand specifying NOP instruction can be appropriately characterized as a variable length instruction". The Official Action further suggests that this "interpretation is supported by the fact that this/these instruction(s) are simultaneously fetched from memory and shown in contiguous locations on Fig. 2A (indicating a single memory location)." Official Action dated 10/9/2007, page 15. However, Sheaffer never indicates that his two separate instructions prior to decoding are a variable length instruction and the Official Action never explains what is meant by the purported variable length instruction with respect to Sheaffer's two separate instructions. The Official Action also never explains how the purported variable length instruction relates to the "expanded width instruction slot used for storing an expanded width instruction" as claimed in claim 9. Sheaffer also never indicates that a single memory location is shown in Fig. 2A. If anything, Fig. 2A indicates a sequence of memory locations from which the contents of each memory location may be individually fetched. As taught by Sheaffer with regard to Fig. 4, "These instructions are preceding and subsequent in terms of program execution order and may or may not be in contiguous memory locations depending on their size, any intervening operand specifiers and/or modifiers, as well as depending on the actual order of program execution." Sheaffer, col. 8, lines 21-26. Thus, Sheaffer's instructions may not even be in contiguous memory locations.

It is further noted that Sheaffer's decode module 205 associates NOP operands with the first instruction opcode 140 (Opcode A) to be represented by a different Opcode B as described at col. 6 lines 14-18. Sheaffer also indicates that "Opcode B may have a higher operand carrying capacity than Opcode A since it is a translation of Opcode A into a different instruction set". Sheaffer, col. 6, lines 18-20. By way of contrast, Fig. 6B of the present specification shows an expanded width instruction in the form of a 40-bit encoding 610 of an expanded MAU instruction. Another exemplary expanded width instruction is shown in Fig. 17 with three instruction encodings 1700 in a 64-bit format. Both of these expanded width instructions are single instructions and are not two separate instructions, such as Sheaffer's single instruction and operand bearing NOP instruction that require a decoder to associate and interpret the two separate instructions. It appears Sheaffer is disclosing an approach which relates two separate instructions through a decoder module and is not at all related to the present invention which utilizes expanded width instructions already formatted to have a format wider than standard width instructions for loading into an expanded width instruction slot in a VLIW memory.

Sheaffer also directly executes the translated instruction represented by Opcode B. As noted above with regard to block 245 of Fig. 2B "the new Opcode B is executed by an execution module 215" as described at col. 6, lines 29-33. Sheaffer does not have a "VLIW memory having a plurality of instruction slots .... from which VLIWs may be fetched for execution" as clamed in claim 9. Sheaffer does not have "at least one expanded width instruction slot having a width that is greater than the standard width instruction slots, the expanded width instruction slot used for storing an expanded width instruction having a format that is wider than the standard width instructions" as claimed in claim 9. Sheaffer does not have a VLIW memory that is "configured for loading said at least one expanded width instruction slot with an expanded width instruction" as claimed in claim 9. Thus, Sheaffer does not teach or make obvious these and various other aspects of claim 9.

<u>Claim 18</u>

Whereas claim 9 is concerned with loading an expanded width instruction in a VLIW stored in a VLIW memory, claim 18 is concerned with fetching a VLIW from a VLIW memory, the VLIW having instructions of a different bit width than the width of instructions stored in a separate instruction memory. It is noted that the fetching of a VLIW and the loading of a VLIW are two different operations. As recited in claim 18, the fetching operation involves the use of "an instruction memory holding a plurality of instructions of a first bit width ... having at least one execute VLIW instruction" and "a very long instruction memory having instruction slots for storing instructions of a second bit width wherein the second bit width is different from the first bit width". Also, as claimed in claim 18, the "very long instruction memory holds VLIWs at addressable locations". The VLIWs held at addressable locations have been previously loaded and it is the loaded VLIWs that reside in the very long instruction memory until they are fetched for execution. Furthermore, a VLIW "may be fetched as a result of executing the at least one execute VLIW instruction" as claimed in claim 18. Sheaffer merely executes decoded instructions. As noted with regard to Sheaffer's block 245 of Fig. 2B "the new Opcode B is executed by an execution module 215" as described at col. 6, lines 29-33. Sheaffer does not teach and does not make obvious a "very long instruction memory having instruction slots for storing instructions of a second bit width wherein the second bit width is different from the first bit width" as claimed in claim 18. Sheaffer also does not teach and does not make obvious the "very long instruction memory holds VLIWs at addressable locations that may be fetched as a result of executing the at least one execute VLIW instruction" as claimed in claim 18.

The Official Action notes "that Applicant has failed to address with respect to many claims the interpretation resulting from the prefix bits shown in col. 1, lines 33-43". Official Action dated 10/7/2007 at page 15. Sheaffer's prefix bits relate to the use special opcodes having the meaning defined by the prefix which "indicates that a subsequent value or subsequent values should be decoded differently" as described at col. 1, lines 34-37. Emphasis added. Sheaffer's use of prefixes or modifiers does not change the fact that Sheaffer's disclosed sequence of operation takes separate instructions and decodes them to a different opcode which is then executed. Sheaffer's use of prefixes and modifiers just makes his decoder more complex. See, for example, Sheaffer col. 7, lines 3-7 where Sheaffer indicates that a "modifier indicates to the decode module that the opcode will use operands from the NOP 110". As such, Sheaffer's use of prefixes or modifiers does not teach and do not make obvious any of the pending claims.

Claim 20

Claim 20 claims a "plurality of instruction slots for storing instruction words forming a VLIW, at least one of said plurality of instruction slots having a compressed format for storing a compressed instruction having a narrower instruction format with respect to an instruction format required in a program storage, wherein the VLIW resides at an addressable location in a VLIW memory, the VLIW memory holding VLIWs that may be fetched for execution". Emphasis added. Compressed instructions are defined in the specification as "instruction formats smaller than 32-bits" at page 13, lines 21-23. The compressed instructions can be stored in "VLIW memory slots, as might be useful in a compressed instruction system" as described at page 13, lines 21-23. Fig. 9A illustrates an exemplary dual arrangement of 15-bit instructions 914 and 916 in basic format 900 and described at page 18, lines 9-20. A compressed instruction is an instruction "having a narrower instruction format with respet to an instruction format required in program storage" as claimed in claim 20. Sheaffer makes no mention of a compressed instruction. Sheaffer does not teach and does not make obvious compressed instructions "having a narrower instruction format with respect to an instruction format required in a program storage" as claimed in claim 20.

Claim 21

Claim 21 distinctly claims various aspects of the invention in a similar manner with regard to claims 9 and 18, and therefore, is novel over and is not rendered obvious by Sheaffer at least for the reasons cited above.

Claims 19 and 22-26

Since dependent claims 19 and 22-26 depend from and contain all the limitations of the base claims 18 and 21, respectively, claims 19 and 22-26 distinguish from the references in the same manner as claims 18 and 21, respectively, placing claims 18-19 and 21-26 in order for allowance.

B. Rejections Under Section 103(a)

These rejections are not supported by the relied upon art. 35 U.S.C. § 103 which governs obviousness indicates that "differences between the subject matter sought to be patented and the prior art" are to be assessed based upon "the subject matter as a whole". Analyzing the entirety of each claim, the rejections under 35 U.S.C. § 103 are not supported by the relied upon art as addressed further below. Only after an analysis of the individual references has been made can it then be considered whether it is fair to combine teachings. However, as addressed further below, fairness requires an analysis of failure of others, the lack of recognition of the problem, and must avoid the improper hindsight reconstruction of the present invention. Such an analysis should consider whether the modifications are actually suggested by the references rather than assuming they are obvious. The 35 U.S.C. § 103 rejections made here pick and choose elements from two or three separate references, none of which presents any motivation for making the suggested combination. This approach constitutes impermissible hindsight and must be avoided. As required by 35 U.S.C. § 103, claims must be considered as a whole. When so considered, the present claims are not obvious.

Claim 10, 11, 13, and 15

Claims 10, 11, 13, and 15 were rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer in view of Moller.

With regard to claim 11, the Official Action suggests that Sheaffer discloses an additional bit to extend address register file addressing and cites col. 3, lines 16-19 for support. However, at the cited text, Sheaffer merely discloses an "immediate operand may be directly provided in a code sequence adjacent to the opcode or the other operands of the instruction".

Storing an operand in line with the code does create an expanded width instruction as claimed. As Sheaffer further notes "[t]hus, an operand may be found directly in line in the code sequence with the instruction and its other operands" at col. 3, lines 27-29. Further, Sheaffer does not teach and does not make obvious the "expanded width instruction slot used for storing an expanded width instruction having a format that is wider than the standard width instructions" and "wherein the store unit instruction slot is an expanded width instruction slot that stores an expanded width store instruction" as claimed in claim 10.

With regard to claim 13, the Official Action suggests that Sheaffer at col. 9, lines 13-19 discloses "that the result is stored in a double-wide register. Since this is treated as one register, this would suggest that loading would work the same way, extending the register addressing". Official Action dated 10/7/2007 at page 10. However, Shaeffer at col. 9, lines 12-15 indicates that "two destination specifiers are used to designate where to store results" which indicates that Sheaffer does not extend the register file addressing no matter how Sheaffer may treat the results. Sheaffer does not teach and does not make obvious these and other aspects of claim 13.

Claims 12, 14, and 16

Claims 12, 14, and 16 were rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer and Moller in view of Tremblay.

With regard to claim 12, the Official Action correctly admits that the combination of Sheaffer and Moller fails to disclose the size of its register file. The Official Action also correctly admits that the combination of Sheaffer and Moller fails to disclose a register file of size 128x32 bit / 64x64 bit / 32x128 bit and relies on Tremblay to solve the admitted deficiency. Official Action further suggests that 32x32 bit / 16x64 bit configurable register file is disclosed by element 103 with Tremblay. However, in searching Sheaffer, Moller, and Tremblay neither a "32x32 bit / 16x64 bit configurable register file" nor a "128x32 bit / 64x64 bit / 32x128 bit configurable register file" as claimed in claim 12 could be found. Tremblay merely discloses a single register file with 128 32-bit registers, at col. 5, lines 39-41. Tremblay does not disclose a "32x32 bit / 16x64 bit configurable register file" nor a "128x32 bit / 64x64 bit/32x128 bit configurable register file" as claimed. The mere disclosure of a register file of any one particular size does not teach and does not make obvious a configurable register file supporting, for example, the "128x32 bit / 64x64 bit/32x128 bit configurable register file" as claimed in claim 12. In particular, claim 12 recites:

> The memory system of claim 11 wherein said expanded width store instruction supports expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size.

The hypothetical combination of Tremblay with Moller and Sheaffer does not create a configurable register file or an expanded width store instruction which "supports expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size" as claimed in claim 12. Rather, the combination would merely support a register file of 128 32-bit registers since the Official Action correctly admited that Sheaffer and Moller fails to disclose the size of its register file. Tremblay does not resolve the admitted deficiencies of the combination of Sheaffer and Moller.

Regarding claims 14 and 16, the Official Action incorporates the rationale of the rejection of claims 12. As noted above the rejection of claim 12 is faulty. Claims 14 and 16 distinguish from the combination of Sheaffer, Moller, and Tremblay in a similar manner to claim 12 addressed above.

Claim 17

Claim 17 was rejected under 35 U.S.C. § 103(a) as unpatentable over Sheaffer in view of Pechanek.

The Official Action suggests that Sheaffer discloses a VLIW memory (VIM) basket (VIMB) at col. 7, lines 1-3. However, at the cited text Sheaffer merely discloses that "the modifier 305 immediately precedes the opcode 100 in the stream of instructions as stored in sequentially fetched memory addresses". Emphasis added. Sheaffer's instructions stored in sequentially fetched memory addresses do not meet the claimed "VLIW having at least one instruction slot that is an expanded instruction slot" and the "VIMB comprising a plurality of VLIWs from which VLIWs may be fetched for execution" as claimed in claim 17.

The Official Action correctly admits that Sheaffer fails to disclose a load indirect instruction or a mask bit field specifying which slots will be loaded in a VLIW having at least one instruction slot that is an expanded instruction slot, the VLIW accessible to be loaded at an addressable location into the VIMB. Pechanek is relied upon to resolve the admitted deficiencies. At the cited text of Pechanek, an XV1 instruction is described which contains "Mask-Enable-bits which select the instructions from the read-out VLIW that are to be scheduled for execution". Pechanek's Mask-Enable-bits which select instructions for read-out are not the same as what is claimed. Claim 17 claims a "load indirect VLIW (LIV) instruction comprising a load mask bit field specifying which instruction slots are to be loaded in a VLIW having at least one instruction slot that is an expanded instruction slot". Thus, Pechanek does not cure the admitted deficiencies of Sheaffer.

To sum up, Sheaffer, Moller, Tremblay, and Pechanek do not show and do not suggest a very long instruction word (VLIW) memory system of claim 9, a VLIW instruction memory basket (VIMB) system of claim 17, a VLIW memory system of claims 18 and 20, and a processing apparatus of claim 21. The cited references do not indicate a recognition of the problems addressed by the present invention. Further, the cited references do not describe a system or apparatus which would solve the problems addressed by the present invention. The claims of the present invention are not taught, are not inherent, and are not obvious in light of the art relied upon.

C. The Official Action's Findings of Obviousness are Also Contrary to Law of the Federal Circuit

As shown above, the invention claimed is not suggested by the relied upon prior art.

The Official Action's rejection suggests that the Official Action did not consider and appreciate the claims as a whole. The claims disclose a unique combination with many features and advantages not shown in the art. It appears that the Official Action has oversimplified the claims and then searched the prior art for the constituent parts. Even with the claims as a guide, however, the Official Action did not recreate the claimed invention.

8. Conclusion

The rejection of claims 9-26 should be reversed and the application promptly allowed.

Respectfully submitted,

[illegible]

Peter H. Priest
Reg. No. 30,210
Priest & Goldstein, PLLC
5015 Southpark Drive, Suite 230
Durham, NC 27713
(919) 806-1600

CLAIMS APPENDIX
(Claims Under Appeal)

Claims 1-8 (cancelled)

9. A very long instruction word (VLIW) memory system comprising:

a VLIW memory having a plurality of instruction slots for storing VLIW instruction words at addressable locations in the VLIW memory from which VLIWs may be fetched for execution;

said plurality of instruction slots comprising standard width instruction slots for storing standard width instructions and at least one expanded width instruction slot having a width that is greater than the standard width instruction slots, the expanded width instruction slot used for storing an expanded width instruction having a format that is wider than the standard width instructions; and

the VLIW memory configured for loading said at least one expanded width instruction slot with an expanded width instruction.

10. The memory system of claim 9 wherein the plurality of instruction slots comprise a store unit instruction slot, a load unit instruction slot, an arithmetic logic unit (ALU) instruction slot, a multiply accumulate unit (MAU) instruction slot, and a data store unit (DSU) instruction slot.

11. The memory system of claim 10 wherein the store unit instruction slot is an expanded width instruction slot that stores an expanded width store instruction including additional bits to extend compute register file addressing, an additional bit to extend address register file addressing, an additional bit to expand an opcode field, or an additional bit to extend a conditional field.

12. The memory system of claim 11 wherein said expanded width store instruction supports expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size.

13. The memory system of claim 10 where the load unit instruction slot is an expanded width instruction slot that stores an expanded width load instruction of a first format for load immediate operations including additional bits to extend compute register file addressing, a bit to extend address file register addressing, a bit to extend a conditional field or sixteen bits to extend an immediate field.

14. The memory system of claim 13 wherein said expanded width load instruction supports expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size.

15. The memory system of claim 10 where the ALU, MAU and DSU instruction slots are expanded width instruction slots that store expanded width ALU, MAU, and DSU instructions including additional bits in each operand field to extend compute register file addressing, a bit to extend an opcode field, or a bit to extend a data type field.

16. The memory system of claim 15 wherein said expanded width ALU, MAU, and DSU instructions each supports expansion of a 32 x 32 bit / 16 x 64 bit configurable register file size to a 128 x 32 bit / 64 x 64 bit/ 32 x 128 bit configurable register file size.

17. A very long instruction word (VLIW) instruction memory basket (VIMB) system comprising:

an instruction register for storing a load indirect VLIW (LIV) instruction comprising a load mask bit field specifying which instruction slots are to be loaded in a VLIW having at least one instruction slot that is an expanded instruction slot, the VLIW accessible to be loaded at an addressable location in the VIMB;

an instruction bit organizer for receiving instructions as data and based on the load mask bit field organizing the bits from the data encoded instructions into proper format for loading into the specified instruction slots in the VIMB; and

the VIMB comprising a plurality of VLIWs from which VLIWs may be fetched for execution, wherein the width of the at least one expanded instruction slot is greater than the width of instructions required in program storage.

18. A very long instruction word (VLIW) memory system comprising:

an instruction memory holding a plurality of instructions of a first bit width, the plurality of instructions having at least one execute VLIW instruction; and

a very long instruction memory having instruction slots for storing instructions of a second bit width wherein the second bit width is different from the first bit width and wherein the very long instruction memory holds VLIWs at addressable locations that may be fetched as a result of executing the at least one execute VLIW instruction.

19. The system of claim 18 wherein instructions of the second bit width are stored in a data memory and delivered to the very long instruction memory utilizing a data bus.

20. A very long instruction word (VLIW) memory system comprising:

a plurality of instruction slots for storing instruction words forming a VLIW, at least one of said plurality of instruction slots having a compressed format for storing a compressed instruction having a narrower instruction format with respect to an instruction format required in a program storage, wherein the VLIW resides at an addressable location in a VLIW memory, the VLIW memory holding VLIWs that may be fetched for execution; and

means for loading said at least one compressed format slot with a compressed instruction.

21. A processing apparatus comprising:

a memory for storing a processing apparatus program comprising short instruction words;

an indirect very long instruction word (VLIW) memory comprising a plurality of instruction slots for storing instruction words, at least one of said instruction slots having an instruction format sized according to execution function and operand storage capacity and independent of the size of the short instruction words, wherein the plurality of instruction slots are organized to form a plurality of VLIWs and each VLIW resides at an addressable location in the indirect VLIW memory, the indirect VLIW memory holding VLIWs that may be fetched for execution;

at least one data memory unit storing instruction operands; and

at least two execution units for executing a VLIW fetched from the indirect VLIW memory in response to a short instruction word dispatched from the memory.

22. The processing apparatus of claim 21 wherein the short instruction words comprise K-bits and the instruction format comprises T-bits, wherein $T \neq K$.

23. The processing apparatus of claim 22 wherein the instruction format comprises at least one operand address field of B-bits supporting direct operand addressing.

24. The processing apparatus of claim 23 wherein the data memory unit has a capacity 2^B data values.

25. The processing apparatus of claim 21 wherein at least one of the at least two execution units operate on a slot instruction format and directly access operands from the data memory unit for execution.

26. The processing apparatus of claim 21 wherein the at least two execution units operate as two execution units when executing two VLIW memory slot instructions as specified by a two slot execute indirect VLIW instruction, and wherein the at least two execution units operate as one execution unit when executing one VLIW memory slot instruction as specified by a one slot execute indirect VLIW instruction..

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.

SPECIFICATION REPLACEMENT PAGE APPENDIX

Please replace the paragraph beginning at page 21, line 16, with the following rewritten paragraph:

Fig. 12 illustrates aspects of a DXP2 processor including a VIM Basket 1210, local instruction registers IR1 1212 and IR2 1214, and decode 1216 and execution 1218 stages in a PE. Some of the differences between the Fig. 12 organization and the SLAMDunk1 organization shown in Fig. 5 are as follows: The VIMBasket 1210 consists of either four slots ~~1243~~ 1244 or six slots ~~1244~~ 1243 in Fig. 12 instead of the fixed five slot arrangement in Fig. 5. Two instruction widths are supported in the DXP2 of Fig. 12, a 64-bit instruction type and a 32-bit instruction type, for example 1220 and 1222 are 64-bit instructions and 1224 and 1226 are 32-bit instructions. A 3-bit UAF field 1230 is used to support the six execution units. A new tag field t 1232 is specified for future use. A second memory 1236 for RFI control parameter storage may be accessed in parallel with a VIMBasket 1210 access. No Short Instruction Word (SIW) path bypassing the VIMB is used in the DXP2 since this capability can be obtained through use of the XV's enable bits.